Name → Purnima Kabadwal
~~Course~~ → B.Tech CSE
Batch - 2022 - 2023
Assignment - 1
Subject : Data Analysis and Algorith-ms.

ASSIGNMENT - 1

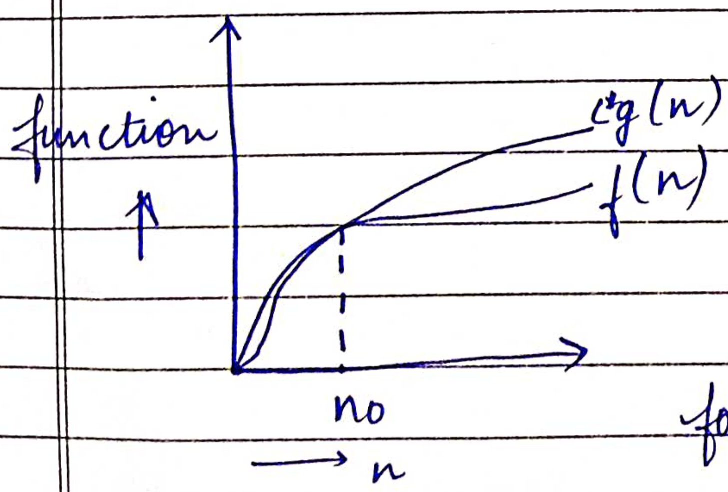
1. What do you understand by Asymptotic notations. Define different Asymtotic notation with examples.

Ans:- Asymptotic means:- Tending to ~~infinity~~ infinity
Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis.
These notations are methamatical tools to represent the complexities

These are 5 complexities:-

1) **Big oH Notation (O)**

Big - oh (O) notation gives an upper bound for a function f(n) to within a constant factor.



$f(n) = O(g(n))$

$f(n) = O(g(n))$ iff $f(n) \le C * g(n)$

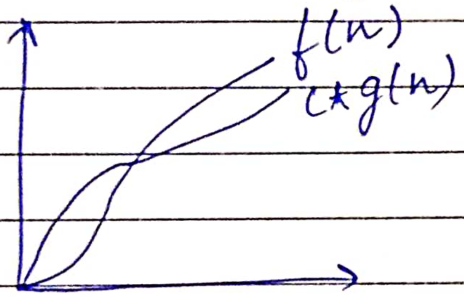
$\forall\ n \geq n_0$

for some constant, $C > 0$

g(n) is tight upper bound of f(n).

2) Big Omega Notation (Ω)

Big omega (Ω) Notation gives a lower bound for a function $f(n)$ to within a constant factor.

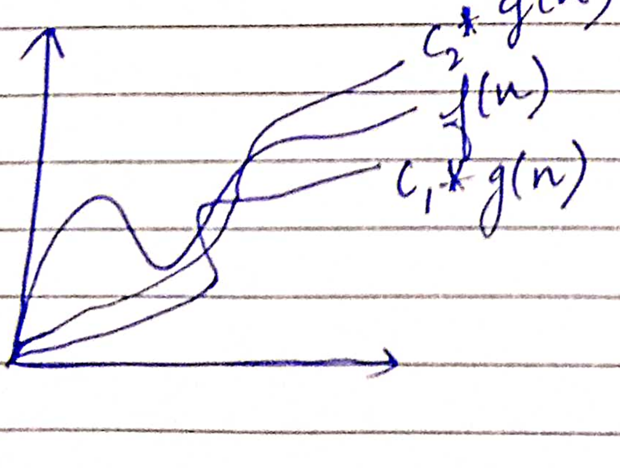


$$f(n) = \Omega(g(n))$$
iff
$$f(n) \geq c * g(n) \quad \forall \; n \geq n_0$$

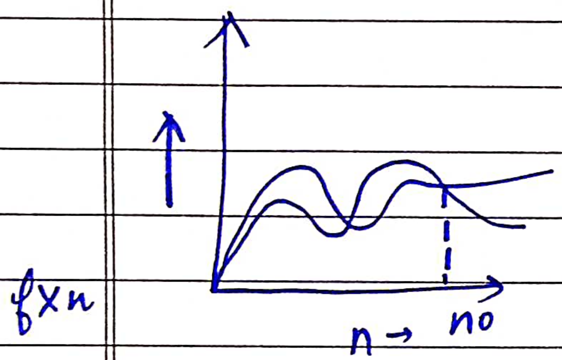
for some constant, $c > 0$.
$g(n)$ is 'tight' lowerbound of $f(n)$.

3) Big theta Notation (Θ)

Big-Theta (Θ) Notation gives bound for a function $f(n)$ within a constant factor.



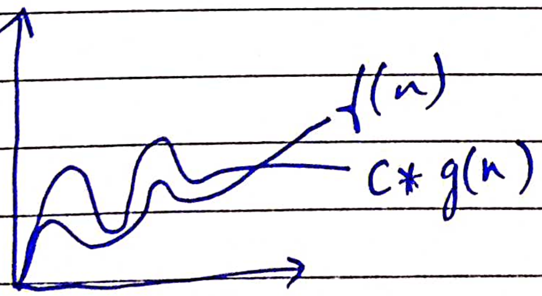
4) Small Theta (θ)



$f(n) = \theta(g(n))$
$g(n)$ is upperbound of $f(n)$
$f(n) = \theta(g(n))$
when
$f(n) < c * g(n)$

$\forall\ n > n_0$
and for all constants, $c > 0$.

5) Small Omega (ω)



$f(n) = \omega(g(n))$
$g(n)$ is lower bound of $f(n)$
when $f(n) > c * g(n)$
$\forall\ n > no$
and for all constants $c > 0$

Date

Question 2.

What should be the time complexity of

```
for i=1 to n — n-times
  {
     i=i*2;
  }
```

Time complexity $= \log_2 n$

The loop executes for n iterations and i gets incremented by a factor of 2.

So, the corresponding values of i will be 1, 2, 4, 16 . . . . n

$$t_k = a r^{k-1}$$

$$n = 1 \times 2^{k-1}$$

$$2n = 2^k$$

$$\log 2n = k \log 2$$

$$\log 2 + \log n = k$$

$$1 + \log n = k$$

$$\boxed{TC = \log_2 n}$$

Mohan

Teacher's Signature

Ques 3 $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$T(n) = 3T(n-1)$ — (1)

putting $n = n-1$ in eqn (1)

$T(n-1) = 3T(n-2)$ — (2)

putting value of $T(n-1)$ from (2) to (1)

$T(n) = 3[3T(n-2)]$

$T(n) = 3^2 T(n-2)$ — (3)

putting $n = n-2$ in eqn (1)

$T(n-2) = 3T(n-3)$ — (4)

putting value of $T(n-2)$ in eqn (3) from (4)

$T(n) = 3^2[3T(n-3)]$ — (5)

$T(n) = 3^3 T[n-3)$

$T(n) = 3T(n-1)\quad 3^2(n-2)\quad 3^3T(n-3)\ \ldots\ 3^n T(n-n$

$T(n) = 3^n T(0)$

$= 3^n$

Ans Time complexity $\theta(3^n)$.

Ques 4 $T(n) = \{ 2T(n-1) - 1$ if $n > 0$, otherwise 1 $\}$ 6

$T(n) = 2T(n-1) - 1$ — (1)

putting value of $T(n-1)$ from (2) to (1)

$T(n) = 2[2T(n-2) - 1] - 1$

putting $n = n-2$ in eq^n (1)

$T(n) = 2^2 T(n-2) - 2^1 - 2^0$ — (3)

putting $n = n-2$ in eq^n (1)

$T(n-2) = 2T(n-3) - 1$ — (4)

putting $T(n-2)$ from (4) to (3)

$T(n) = 2^2[2T(n-3) - 1] - 2^1 - 2^0$

$T(n) = 2^3 T(n-3) - 2^2 - 2^1 - 2^0$ — (5)

$T(n) = 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2 - 2^1 - 2^0$

$= 2^n - (2^n - 1)$

$T(n) = \cancel{2^n - 2^n} + 1$

$T(n) = O(1)$ Ans

Ques 5 What should be the time complexity of

```
int i=1, s=1;
while (s<=n)
{ i++;
  s=s+i;
    printf("#");
}
```

we can define the terms 's' according to relation $s_i = s_{i-1} + i$. The value of 'i' increases by 1 for each iteration

The value contained in 's' at the ith iteration is the sum of the first 'i' positive integers.

Let k be the total number of iterations

while loop terminates if

$$1+2+3+\ldots\ldots+k$$

$$=[k(k+1)/2]>n$$

so $k = O(\sqrt{n})$

Time complexity = $O(\sqrt{n})$

Question 6 Time complexity of :-

```
void function (int n)
{ int i , count = 0 ;
  for (i=1 ; i*k = n ; i++)
    count ++ ;
}
```

$i^2 <= n$

$i <= \sqrt{n}$

$i = 1, 2, 3, 4 \ldots \sqrt{n}$

$\sum_{i=1}^{n} \quad 1+2+3+4 \ldots + \sqrt{n}$

$$T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$$

$$T(n) = \frac{n \times \sqrt{n}}{2}$$

$T(n) = O(n)$ Ans

Question 7 :-

Time complexity of :-

```
void function (int n)
{
  void i, j, k, count=0;
  for (i=n/2; i<=n; i++)
    for (j=1; j<=n; j=j*2)
      for (k=1; k<=n; k=k*2)
  count++;
}
```

for $k = k*2$

$k = 1, 2, 4, 8 \ldots\ldots n$

G.P $\quad a=1 \quad r=2$

$$n = \frac{a(r^n - 1)}{r-1} = \frac{1(2^k - 1)}{1}$$

$$\boxed{n = 2^k}$$

$\log n = k$

i	j	k
1	$\log n$	$\log n \times \log n$
2	$\log n$	$\log n \times \log n$
⋮	⋮	⋮
n	$\log n$	$\log n \times \log n$

$n * \log n * \log n$

$\Rightarrow O(n\log^2 n)$ Ans//

Question 9

The complexity of :-

```
void function (int n)
{ for (i=1 to n)
  {
     for (j=1 ; j<=n ; j=j+1)
       printf ("*");
  }
}
```

Answer:→ for $i=1 \Rightarrow j=1, 2, 3, 4 \ldots\ldots n$ $\quad n$

for $i=2 \Rightarrow j=1, 3, 5 \ldots\ldots n$ $\quad n/2$

$n/3$

$\vdots$

for $i=n \rightarrow j=1 \ldots\ldots\ldots\ldots$ $\quad n$

$$\sum_{j=n}^{1} n+\frac{n}{2}+\frac{n}{3}+\frac{n}{4}+\ldots\ldots+1$$

$$\sum_{j=n}^{1} n\left[1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4} \ldots \frac{1}{n}\right]$$

$n(\log n)$

$T(n) = O(\log n)$

Question 10

For the functions n^k and c^k, what is asymtotic notation between the functio? Assume that k>=1 and c>1 are constants

find out the value of c and no. for which relation holds.

Given :- n^k c^n

$n^k = O(c^n)$

as $n^k \leq a c^n$

$\forall$ $n \geq n_0$ and some constant $a>0$

for $n_0 = 1$

$1^k \leq a^{2^1}$

$n_0 = 1$ and $c = 2$ Ans